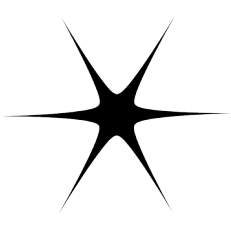


Calcolatrice is a software application that can **execute** **plain calculus** but also **complex formulas** with variables and **retain** the ***history of your work***, so in respect with the standard app provided by the operating system is really useful.

Written in **POWER-KI**, let you use many operators and built-in function of this powerful language, so giving a fast-track to learn coding, in plus it is open source and free.

1.0.0

Calcolatrice

POWER-KI Apps

Productivity

REVISION

MAJOR REVISION HISTORY		CREATED/REVISED			APROVED		
#	NOTE	DATE	BY	NAME	DATE	BY	NAME
0		21/09/20	DTC				

XPLAB s.a.s
viale Sant Eufemia , 39
25135 Brescia – ITALY

Tel. +39 030 2350035

www.XPLAB.net
www.power-ki.com
www.PowerBerry.tech

Summary

Disclaimer

While XPLAB sas make every effort to deliver high quality products, we do not guarantee that our products are free from defects.

Our software is provided "**as is**," and you use the software at your own risk.

We make no warranties as to performance, merchantability, fitness for a particular purpose, or any other warranties whether expressed or implied.
No oral or written communication from or information provided by XPLAB sas shall create a warranty.

Under no circumstances shall XPLAB sas be liable for direct, indirect, special, incidental, or consequential damages resulting from the use, misuse, or inability to use this software, even if XPLAB sas has been advised of the possibility of such damages.

Document Information

Summary

Describe the POWER-KI app Calcolatrice.

Purpose

Help users.

References

[1] POWER-KI : a programming language - PRELUDIO
(C)2012 XPLAB - BRESCIA - ITALY
C.A. PERANI
ISBN 978-88-907392-1-7

[2]

Document Change

Terms and Definition

Glossary entry	Entry definition
PWK	POWER-KI
NV	Numerical value (like 10)
NNV	Not Numerica Value (char strings like Italy)

1 User interface

The use is very simple:

insert your calculus and press <ENTER> or the button [EXECUTE];

the result is returned in the result box and in the History grid both the calculus and the result are added on top (the last 50, unic computations).

From History grid you can reload (and re-execute) a calculus or delete a line, or flush all.

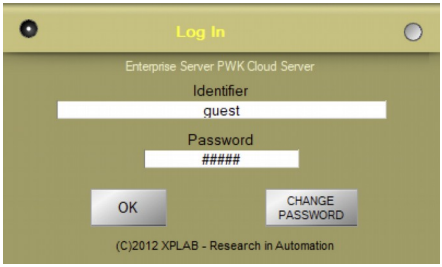
With the [X] (erase) button or with <ESC>, inside the input box, the input box is cleared.

You can define Formulas with Parameters. When you enter a symbol with § (paragraph) as first character, if it does not already exists is added to the parameter grid and you can insert the values to be considered for the execution in the VALUE column.

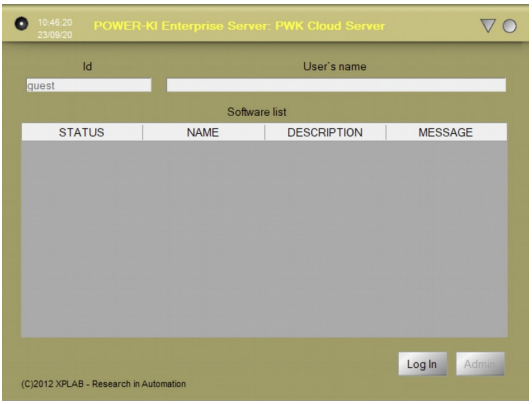
The content of the Input box and the relative parameters can be saved with name in the formula repository and its name will be added on the Formula grid for a later use (mouse double click or [LOAD]).

With the button [Goto: www.POWER-Ki.com] two tab are opened on your browser: site main page and a Google search for info about POWER-KI and XPLAB.

With the button [Goto: Power CLOUD] you get the access to PWK-EnterpriseServer for CLOUD services.



Id and Password are ***guest***.



After the login from the menù you can download update, new applications, access CLOUD services etc.

2 How to use

In the Input box you can insert simple calculus, but also thanks to POWER-KI, complex formulas or even mini programs.

Key Points	example
Symbol beginning with § (paragraph) are parameter of the formula. (NOTE: this is a specific *Calcolatrice* Notation)	10+§H
Symbol beginning with **°** are Temporary variable	°a
Symbol beginning and ending with **°** are Static variables, that retains its value.	°b°
Symbol beginning with £ are TEXT	£MyName
The separator of Float number is **.** (dot)	3.14
The , (comma) separate parameters	FLT(§R,2)
; (Semi colon) separate statements and clear the stack	°a=10;°a +2
0x if the prefix for hexadecimal number	**0x**abAB
0b is the prefix for binary number	**0b**010111

2.1 by Examples

2.1.1 Math

+ - * /
Input box
10*2
Result Box
20

Constants
Input box
PI..ENP
Result Box
3.141592653589793 2.718281828459045

PI is: Greek P 3.14..
ENP is: e nepherian

The ".." is an operator that le concatenate symbol with a space between, "++" does the same thing but without the space.

SIN COS TAN SINH COSH TANH ASIN ACOS ATAN ATAN2

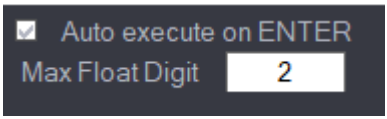
Input box

```
sin(pi/4)
```

Result Box

```
0.71
```

For Single Float result the default number of float digit can be set



setting Max Float Digit to 0 or in any case for multiple results, let you use:

FLT

Input box

```
FLT(sin(pi/4),4)
```

Result Box

```
0.7071
```

HEX and BIN

Input box

```
hex(125)..bin(125)..0b10+0x10
```

Result Box

```
0x7d 0b1111101 18
```

Calcolatrice accept number also in hexadecimal (0x...) and binary (0b...) form.

2.1.2 Temporary variables

You can define Temporary variable (that retain a value only inside the calculus) with symbol beginning with ° (like: **°myVar**).

Having this:

Input box

```
FLT(sin(pi/4),2)..FLT(cos(pi/
4),2)..FLT(tan(pi/4),2)
```

Result Box

```
0.71 0.71 1.00
```

we can use **°A** :

Input box

```
°A=pi/4;FLT(sin(°A),2)..
FLT(cos(°A),2)..FLT(tan(°A),2)..
flt(°a,4)
```

Result Box

```
0.71 0.71 1.00 1.0472
```

in this example we also use ; (semi-colon) to separate statements, the effect of the semi-colon is also to clear pending values (clear the stack).

As you can note symbol are case insensitive: °a = °A , FLT = flt

2.1.3 Static variables

Static variable are retentive between executions and are that defined with a ° at the beginning and at the end, like **°myPvar°** .
So we can broke the example in two executions:

Input box

```
°A=pi/4;°a
```

Result Box

```
1.0472
```

and the we can use **°A°** in any subsequent calculus:

Input box

```
FLT(sin(°A°),2)..
FLT(cos(°A°),2)..FLT(tan(°A°),2)
```

Result Box

```
0.71 0.71 1.00
```

2.1.4 Formulas

Calcolatrice let define formulas with parameters, these are defined prepending **§** (paragrath**)** at the beginning, like: **§myPar**

Entering in the Input box a symbol that begin with § the Parameters grid if not, is displayed and if the parameter is not already exists, is added.

In this way is very simple re-execute a formula with different values.

Input box
SQRT(§cat1^2+§cat2^2)

§PARAMETERS	VALUE
§cat1	3
§cat2	4

Result Box
5

Any content of the Input box can be saved as a Formula also if it does non have parameters.

2.1.5 Iteration

in a formula can be inserted iterations, the schema is;

```
#WHILE(cond);
        do someting;
#END;
```

The iteration can be shortened with `#SKIP;` or ended with `#BREAK;`

As example the Fibonacci series:

Input box

```
°r=0; °a=0; °b=1;
#WHILE(°a<§n);
°b=(°a+°b,°b)->°a;
°r=°r++","++°a;
#END;
°r
```

§PARAMETERS	VALUE
§n	4

Result Box

```
0,1,1,2,2,5
```

2.1.6 Check condition

Two forms are possible.

The first is a flow control:

```
#IF(cond);
        do something;
#ELSE;
        something else;
#END;
```

Input box

```
#IF(§n>100);
°r=§n-100;
#else;
°r=100-§n;
#END;
°r
```

§PARAMETERS	VALUE
§n	25

Result Box

75

The second is a function:

Input box
°r=if(§a<0,§a*-1,§a);°r

§PARAMETERS	VALUE
§a	-5

Result Box
5

2.1.7 Not only numbers

In Calculus you can manage also string of characters:

Input box

```
if(§a>100,£above,£below)
```

§PARAMETERS	VALUE
§a	101

Result Box

above

If the string does not contains spaces can be inserted prepending £ (literal), otherwise with double quote: `"text string"`.

SPLT

Input box

```
splt("xplab.net",".",£LEFT)
```

Result Box

xplab

2.1.8 For advanced Users

In PWK there are many LIBs that offer functions for various tasks, just as an example:

FS_FIND

Input box

```
°L=fs_find("*.*");
°F=cat(LIS_use(°L,crlf));
trash(°L);
°F
```

Result Box

(list of files in the current directory)

Explanation:

`°L=fs_find("*.*");`
search all (*.*) the files and directory in the current directory and return a LIS;

`°F=cat(LIS_use(°L,crlf));`
the found elements are concatenated in a string (°F), separated by CR LF;

`trash(°L)`
the pointer of the LIS is destroyed;

`°F`
is returned to be displayed .

3 Operators and functions overview

In PWK all mathematic is performed on F64 (float 64 bit) values.
The complete reference is available in POWER-KI manuals see:
http://www.POWER-KI.com
https://github.com/POWER-KI/POWER-KI

Operators
£, (), ~, =, +=, -=, //, /, +, -, *, ==, !=, <, >, <=, >=, &, |, %, ^, !, ->, <<, >>, <<=, ++, .. , NOT, AND, XOR, OR, ZNOT, ZAND, ZOR, ZXOR, ZSUM, ZFSUM, PAND, POR, PXOR, SIN, COS, TAN, SINH, COSH, TANH, ASIN, ACOS, ATAN, ATAN2, SQRT, EXP, LOG, LOG10, MOD, ABS

```
£ (litteral)
------------------------
prepended defines a constant:
£test; !! is equivalent to write; "test"

~ (tilde) (note: from keyboard obtainable by pressing ALT+126, keep
ALT and press sequentially 1,2,6)
------------------------
applied to an attribute, it returns true (1), if the attribute
content is valid, that is not null and not only composed by spaces:
IF( ~v1, £full, £empty); !! the opposite is; IF( ~v1==0, £empty,
£full);

+= and -= (addiction and subtraction unary operators)
------------------------
THREADs are used a lot in PWK programs. There are situations that
require atomic operations. Suppose we want to increase the value of
a variable: A = A + 1 in a situation of parallelism it could happen
that between the acquisition of the value of A its increase and its
reassignment another Thread has changed its value, the unary
```


```
operators guarantee the atomic nature of the operation.

/ (division)
------------------------
What is the difference with the usual division?
The division by ZERO results ZERO!

// (integer division)
------------------------
10.5 // 3 = 3

^ (exponentiation)
------------------------
3 ^ 2 = 9

== CMP (comparation)
------------------------
ZERO is not NULL:
r=if( 0 == NULL, £TRUE, £FALSE); !! r results FALSE;
For the NNV (not numerical values), the '==' operator executes a
case insensitive comparation, ignoring the spaces at the beggining
and at the end, if you need an accurate 'char by char' and case
comparision, you have to use CMP:
s1="first";
s2=" first ";
s3=" First";

c1=if( s1==s2,£YES,£NO); !! c1 results YES;
c2=if( s1 cmp s2 == 0,£YES,£NO); !! c2 results NO;
c3=if( s1==s3,£YES,£NO);!! c3 results YES;

& | % ! (and or xor not)
------------------------
Binary Operators.

AND OR XOR NOT
------------------------
LOGIC operations.

<< >> (Left and Right Shift)
------------------------
```

```
Shift operators: their behaviour changes with NV or NNV values:
a=0b1101; b= a >> 2; !! b results 0b11;
A="Tested"; B= A >>2; !! B results "Test";

++ <<= (concatenation)
------------------------
A="Test"; B="One"; C= A ++ B; !! C results "TestOne";
A="Test"; B="One"; C= A <<= B; !! C results "TestOne";

.. (concatenation with space)
------------------------
A="Test"; B="One"; C= A .. B; !! C results "Test One";

ZNOT ZAND ZOR ZXOR ZSUM (Fuzzy operators)
------------------------
Note that ZSUM is an extension of PWK and for which there is also a
form as a function ZFSUM that can "add" more than one element, both
are drawn on the error (0 = no error) true (1) the result goes
inverted (r = a ZSUM b; r = 1-r;);

PAND POR PXOR (probabilistic operators)
------------------------
To execute probabilistic operations.
REF, EXIST, TYPOF (references)
------------------------
REF is an indirect reference to an item or an attribute, that can
be used as element left or right in the assignations;
EXIST verifies the existance of an item or an attribute;
TYPOF returns the type of an ITEM or an attribute.

HEX, BIN, FLT, INT, UNS, CHAR, CHARCOD (conversions)
------------------------
In general, the use is intuitive.
FLT allows to specify the decimal number of a value,
CHAR encodes one or more NVs in a symbol,
CHARCOD returns the numeric code of a character.

MIN, MAX, LIM (check on NVs)
------------------------
MIN returns the minimum of a list of values;
MAX returns the maximum of a list of values;
```


```
LIM(v, mn, mx), if v exceeds the limits, returns a value between
them.

IF (ternary)
------------------------
IF(c, a, b), as function, evalues the first parameter and, if true,
return the second value (a), else the third (b). As a function, all
the parameters are evalued before the assignation.

ISNULL, ISEMPTY, ISTRUE, ISERR, TSTX, ISNUM, ISFLT
------------------------
Test function.
ISTRUE(x)
- if x is a NV, it returns true (1) se x>0;
- if x is a NNV, it returns true (1), if x is not null or x not
contains only space characters;
ISEMPTY(x) returns true(1) if x is empty and not NULL;
ISNULL(x) returns true (1), if x is NULL;
ISERR(x) returns true (1), if x is NULL or <0;
TSTX(x, "One,Two") returns £One, if x is true, else £Two
ISNUM(X) returns true (1), if x is a NV
ISFLT(x) returns true (1), if x is a NV of FLT (float) type.

BITF, MID (extraction and/or modify of a symbol)
------------------------
For NVs, you can use BITF to obtain or extract single or multi bit
values from a source value.
For NNVs, you can use MID to obtain or modify a part of a symbol.

LEN, NSP, FST, LST, CAT, LWR, UPR, RTF2TXT, NNL, NNLv (symbol
operation)
------------------------
LEN(s) returns the length of s in characters;
NSP(s) deletes from s the space characters at the beginning and at
the end;
FST(s) returns or substitutes the first character of s;
LST(s) returns or subtitutes the last character of s;
LWR(s) transforms all the characters of s in the lower case
version;
UPR(s) transforms all the characters of s in the upper case
version;
```

```
CAT concatenates symbols;
RTF2TXT(x) converts a RTF text in unicode.
NNLv(x,v) , if x is null, returns the value of the second parameter
or an empty symbol;
NNL, if x is null, returns an empty symbol.

FRMT (format)
------------------------
Create a symbol using the C printf notation,

SPLT, TKNZ, TKNZOP, CSV, CSVTBL (split symbols)
------------------------
SPTL extracts a side (Left/Right) by a symbol, related to a
separator;
TKNZ/TKNZOP returns a LIS, obtained splitting a symbol, using a
list of separators;
CSV returns a LIS, obtained splitting a symbol, using a separator
(,);
CSVTBL returns a TBL, obtained splitting a symbol, using different
separators for items and rows, preserving the values contained into
blocks.

SRCH, MTCH (search into symbols)
------------------------
SRCH searches a symbol inside an other (with various parameters);
MTCH comparate symbols for similarity.

QUOS, QUOD,QUOSE, QUODE, ESCP (incapsulation of symbols with
quotes)
------------------------
QUOS (single quote) encapsulates a symbol with single quotes;
QUOD (double quote) encapsulates a symbol with double quotes;
QUOSE, QUODE as above but preserves (doubling it) the eventual
encapsulation character contained in the symbol.

TMR, CLOCK (the time)
------------------------
TMR obtains the system time (from the start of the program or
thread) in ms or at high resolution, with possible comparison with
a given value;
CLOCK returns the number of seconds elapsed since january 1st 1970.
```


```
OSEXEC, OSSHELL, OSSTART, PWKTASK (lauch Operating System processes
or commands)
------------------------
OSEXEC executes a command on the O.S. (synchronous);
OSSHELL executes an O.S. command;
OSSTART launches a new process;
PWKTASK starts a PWK program as a process.


PTRTYP, PTRLIS, PTRDUP, THASH (the PTRs, pointers)
------------------------
PTRTYP returns informations about a PTR;
PTRLIS returns the list or the allocated pointers;
PTRDUP duplicates a pointer,
TRASH is used to delete pointers no longer in use .

CRLF, BOM, UID, UCNT, PI, ENP, RAND (costants or almost)
------------------------
CRLF codificates the end-line symbol (0x0d0A);
BOM ByteOrderMark ( 0xFEFF) for UTF-16;
UID generates unique ids, within the PWK application;
UCNT is a unique counter, within the PWK application;
PI Greek P 3.14..
ENP e nepherian
RAND returns a randomic value.
```

4 Where Next

Calcolatrice is an Open Source and Free application written in POWER-KI,

If you are interested in coding you can consider to learn POWER-KI,

Using ***Calcolatrice*** you have already learned some basic elements, with a little effort, you can go ahead and write your own application for a vast range of fields:

- IoT (Internet Of Things),
- account,
- management,
- industrial,
- web ...